Visualization of Fluid Flows using *Mathematica*

James E. Coleman
Midshipman
U. S. Naval Academy
Annapolis MD, 21412

Reza Malek–Madani
Department of Mathematics
U. S. Naval Academy
Annapolis, MD 21402

David R. Smith
Department of Oceanography
U. S. Naval Academy
Annapolis, MD 21402

Abstract

We present numerical solutions to two fundamental flows in fluid dynamics: a) Serrin's swirling vortex, and b) the Rayleigh–Bénard flow In each case we solve a system of nonlinear ordinary differential equations that arise from the velocity field and monitor the evolution of particles of fluids. The velocity fields in turn are solutions of partial differential equations that govern the conservation of mass and the balance of linear momentum in each flow. All of the computations, whether symbolic or numerical, are performed in *Mathematica*. The computational code essential to visualizing each flow is presented in the sequel.

1 Introduction

Data visualization has always played an important role in the mathematical analysis of nonlinear partial and ordinary differential equations. It is often difficult, if not impossible, to obtain exact and analytic solutions of these equations and, as a result, one either resorts to approximate methods or applies analytic techniques in order to gain insight into the qualitative behavior of solutions. In the context of fluid dynamics, where one is confronted with the analysis of the Navier–Stokes equations, a visual rendition of an approximate solution and its comparison with the physical setting one is modeling offers an extra measure of validation. It is not uncommon, for instance, to discover that one should look more carefully at the stability of a certain solution after viewing the dynamics of fluid particles rendered by the mathematical model.

The mathematical models we discuss in this paper have fundamental importance in fluid dynamics and ocean–atmosphere modeling. The techniques we describe in solving the underlying differential equations are carried out in the software package *Mathematica* on a standard personal computer. We are able to use internal functions of this software to solve nonlinear boundary-value and initial–value problems that arise from considering special solutions of the Navier–Stokes equations. The techniques that we describe may be combined with the Galerkin scheme to allow for studying physical problems with less symmetry in their solution structure as well as more complicated boundary conditions considered here.

2 Serrin's Swirling Vortex

Serrin's swirling vortex is a steady–state solution to the Navier–Stokes equations. It describes the dynamic flow in a system involving a vortex line interacting with a perpendicular boundary surface to which fluid particles adhere. This solution provides a model of the idealized fluid flow of an atmospheric tornado in contact with the Earth's surface. The velocity–pressure pair, $(\mathbf{v}, p)$, of this solution satisfies the steady–state Navier–Stokes equations

$$\rho\,(\mathbf{v}\cdot\nabla\mathbf{v}) = -\nabla p + \mu\triangle\mathbf{v}, \qquad \text{div }\mathbf{v} = 0, \quad (1)$$

where μ is the kinematic viscosity. The first set of equations represents the balance of linear momentum while the second equation expresses the conservation of mass. Equations (1) are supplemented by the boundary conditions

$$\mathbf{v}|_{z=0} = 0, \qquad \lim_{||\mathbf{x}||\to\infty} \mathbf{v} = 0, \qquad (2)$$

The second boundary condition (2) is imposed since we seek solutions whose behavior is local in space.

Let r, α, and θ denote the standard spherical coordinates. The domain of the flow is $r > 0$, $0 \le \alpha < \frac{\pi}{2}$, and $0 \le \theta < 2\pi$. Let v_r, v_α, and v_θ denote the components of the velocity vector $\mathbf{v}$ in this coordinate system. Following Serrin [1], we seek solutions to (1)–(2) with the special structure

$$v_r = \frac{G(\cos\alpha)}{r\sin\alpha}, \quad v_\alpha = \frac{F(\cos\alpha)}{r\sin\alpha}, \quad \text{and}$$

$$v_\theta = \frac{\Omega(\cos\alpha)}{r\sin\alpha}. \qquad (3)$$

After transforming equations (1)–(2) to spherical coordinates and substituting (3) there, we find that $G = F'\sin\alpha$ and that F and Ω must satisfy the following system of ordinary differential equations: Let f be related to F through the relation $F = \frac{1}{k}(1-x^2)f$ where

$$k = \frac{\rho}{2\mu}.$$

Let Q be defined by

$$Q(x) = 2(1-x^2)\int_0^x \frac{t\Omega^2(t)}{(1-t^2)^2}\,dt + 2x\int_x^1 \frac{\Omega^2(t)}{(1+t)^2}\,dt - (x-x^2)P,$$

where P is a contant of integration. Then f and Ω satisfy the system of integro–differential equations

$$f' + f^2 = \frac{k^2}{(1-x^2)^2}Q, \quad \Omega'' + 2f\Omega' = 0, \quad (4)$$

with $x \in (0,1)$. This system is complemented with the boundary conditions

$$f(0) = \Omega(0) = 0, \quad \Omega(1) = 1. \qquad (5)$$

We have skipped quite a few steps in transforming (1)–(2) to (4)–(5). The computations in these steps are generally well–known and straightforward. A reader interested in the details of this analysis is strongly encouraged to consult Serrin [1].

We find an approximate solution to the boundary–value problem in (4)–(5) by applying the Picard iteration scheme together with the shooting method to this system (see Malek–Madani [2] for details of how one implements these techniques in *Mathematica*). The Picard iteration scheme is used to reduce the system of integro–differential equations to a system of ordinary differential equation; The latter system can then be solved accurately with routines that already exist in *Mathematica.* The shooting method is employed to convert the boundary–value problem to a sequence of initial–value problems. The internal functions `NDSolve` and `FindRoot` of *Mathematica* are combined at this stage to obtain a sequence of solutions to appropriate initial–value problems that ultimately converge to the solution of the boundary–value problem. Program 1 displays the syntax of the *Mathematica* commands used in implementing the Picard iteration scheme and the shooting method.

Program 1:

```
P = 1; k = 1;
llabel=StringJoin["P = ",
          ToString[P],",
         k = ", ToString[k]];
xfinal=.9999;
y2[x_]=x;
Do[
 sol1=NDSolve[
  {y1'[x]==k^2/(1-x^2)^2 *
  (2(1-x)^2* NIntegrate[
  t y2[t]^2/(1-t^2)^2, {t,0,x},
    MaxRecursion->10]+
  2x NIntegrate[y2[t]^2/(1+t)^2,
  {t,x,xfinal},
    MaxRecursion->10]-
  P(x-x^2))-y1[x]^2, y1[0]==0},
   y1,{x, 0, xfinal}];
 Clear[y2];
   yone[x_] = First[y1[x] /. sol1];
 output=FindRoot[First[
   Evaluate[y2[xfinal] /.
    NDSolve[{y2'[x]==y3[x],
    y3'[x]==-2 yone[x] y3[x],
    y2[0]==0,y3[0]==a},{y2,y3},
    {x,0,xfinal}]]]-1,
    {a, 0.1, 0.9}];
 sol2=NDSolve[{y2'[x]==y3[x],
    y3'[x]==-2 yone[x] y3[x],
    y2[0]==0, y3[0]==a /. output},
    {y2,y3},
    {x,0,xfinal}, MaxSteps->2000,
      WorkingPrecision->15];
    y2[x_]=First[y2[x] /.sol2];
    Print[y2[0.1]],
{i,1,10}];
F1[x_]=1/k (1-x^2) yone[x];
F2[x_]=D[F1[x],x] Sqrt[1-x^2];
Omg1[x_]=First[First[y2[x]/.sol2]];
graph1=Plot[
 {F1[x],F2[x],Omg1[x]},{x,0,xfinal},
 PlotRange->All, PlotLabel->llabel];
```

Figure 1 shows the output of this program, which consists of the graphs of F, G, and Ω.

We now proceed with obtaining the necessary data to visualize the fluid flow generated by the triple (F, G, Ω). This triple defines the components of the velocity field $\mathbf{v}$ through the relations (3). The dynamical system that governs the evolution of particles of fluids under this velocity field is a set of three nonlinear ordinary differential equations for the triple $(r(t), \alpha(t), \theta(t))$. The rate of change of r, α, and θ with respect to t are related to v_r, v_α, and v_θ through the formulas

$$\frac{dr}{dt} = \frac{G(\cos\alpha)}{r}, \quad \frac{d\theta}{dt} = \frac{\Omega(\cos\alpha)}{r^2 \sin^2\alpha}, \quad \text{and}$$

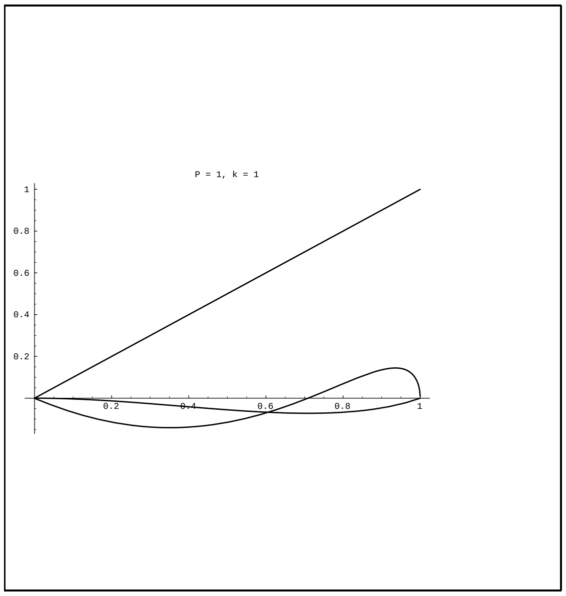

Figure 1: The graphs of F, G, and Ω when $P = 1$ and $k = 1$.

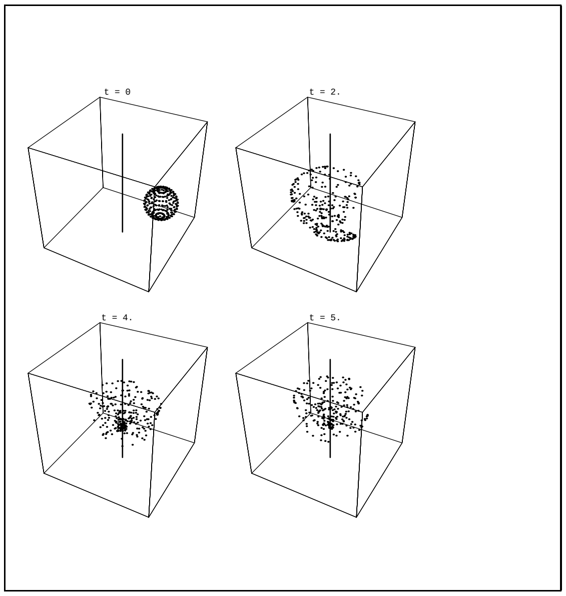

Figure 2: Four snapshots of a parcel of fluid undergoing a tornado–like motion. Here, a parcel of fluid consisting of 325 data points that originally occupy a sphere of radius 0.3 near the vortex filament is considered. The evolution of this parcel is then displayed at 51 equal subsequent time intervals, four of which are shown in this figure.

$$\frac{d\alpha}{dt} = \frac{F(\cos\alpha)}{r^2 \sin\alpha}. \tag{6}$$

Program 2 starts with the output of Program 1, namely, the numerical values of F, G, and Ω and proceeds to solve for the solutions of a typical initial–value problem based on (6). The program is written so that one could visually follow the evolution of any number of parcels of fluid, each consisting of any number of particles. The initial shape of each parcel is taken to be a sphere. Each parcel is colored differently so that one could observe mixing of different portions of the atmosphere due to the action of the dynamical system. The syntax of this program is as follows:

Program 2:

```
time=5;
tinterval=.1;
parcels=1;
Va[alpha_, theta_, r_]=
  (F1[Cos[alpha]])/(r Sin[alpha]);
Vr[alpha_, theta_, r_]=
  (F2[Cos[alpha]])/(r);
Vt[alpha_, theta_, r_]=
  (Omg1[Cos[alpha]])/(r Sin[alpha]);
diffeqn[tfinal_, {b_, c_, d_}]:=
 NDSolve[{alpha'[t]==(1/r[t])*
  Va[alpha[t], theta[t], r[t]],
  theta'[t]==1/(r[t]Sin[alpha[t]])*
  Vt[alpha[t], theta[t], r[t]],
 r'[t]==Vr[alpha[t],
     theta[t], r[t]],
  alpha[0]==b, theta[0]==c,
    r[0]==d},
   {alpha, theta, r}, {t,0,tfinal}];
e=.5;
f=.5;
g=.5;
h=.3;
n=12; colors={RGBColor[1,0,0],
 RGBColor[0,1,0], RGBColor[0,0,1]};
Print["Parcels being generated ..."];
filament=Graphics3D[
     Line[{{0,0,0},{0,0,2}}]];
Do[
 CoordPointsCart[m+1]=
   Flatten[Table[{
    x->N[e+m+h Sin[v] Sin[w]],
     y->N[f+h Sin[v] Cos[w]],
      z->N[g+h Cos[v]]},
       {w, 0,2 Pi, Pi/n},
         {v, 0, Pi, Pi/n}],1];
 PointsCart[m+1]=
  {x,y,z}/. CoordPointsCart[m+1];
 MakePointsCart[m+1]=
  Table[Point[PointsCart[m+1][[i]]],
      {i, Length[PointsCart[m+1]]}];
 graph[m+1] = Graphics3D[
   {colors[[m+1]],
     MakePointsCart[m+1]}],
{m,0, parcels-1}];
snapshot[1] = Join[{filament},
   Table[graph[m+1],
  {m, 0, parcels-1}]];
Print["Parcels generated"];
Print["Generating new snapshots"];
plot[1]=Show[snapshot[1],
  PlotRange->{{-1,1},{-1,1},{0,2}},
    PlotLabel->"t = 0"];
Do[SpherePoints[m+1]=
   {N[ArcCos[z/Sqrt[x^2+y^2+z^2]]],
     N[ArcCos[y/Sqrt[x^2+y^2]]],
      N[Sqrt[x^2+y^2+z^2]]} /.
      CoordPointsCart[m+1],
{m,0,parcels-1}];
howlong=Timing[
   Do[oldsolution[m+1]=
     Flatten[Table[
      diffeqn[time,
       SpherePoints[m+1][[i]]],
         {i,1,Length[
   SpherePoints[m+1]]}],1],
{m, 0, parcels-1}]];
Print["Finished with NDSolve ..."];
Print["CPU used ...",howlong];
Do[currentTime=j*tinterval;
  Do[cartsol[m+1]=Table[{
   r[t] Sin[alpha[t]] Sin[theta[t]],
   r[t] Sin[alpha[t]] Cos[theta[t]],
   r[t] Cos[alpha[t]]} /.
   oldsolution[m+1][[i]]/.
   t->currentTime,
   {i,1, Length[oldsolution[m+1]]}],
{m,0,parcels-1}];
Do[points2[m+1] = Table[
  Point[cartsol[m+1][[i]]],
   {i, Length[cartsol[m+1]]}],
{m,0,parcels-1}];
Do[graph[m+1]=Graphics3D[
 {colors[[m+1]], points2[m+1]}],
{m,0,parcels-1}];
snapshot[j+1] = Join[{filament},
 Table[graph[m+1],
   {m, 0,parcels-1}]];
label=StringJoin["t = ",
  ToString[j*tinterval]];
plot[j+1]=Show[snapshot[j+1],
  PlotRange->{{-1,1},{-1,1},{0,2}},
  PlotLabel->label],
{j, 1, 50}];
graph2=Show[GraphicsArray[
 {{plot[1], plot[21]},
  {plot[41], plot[51]}}]];
```

Figure 2 shows a portion of the output of this program. The entire output consists of 51 frames displaying the position of 325 particles at various times $t > 0$; These particles formed a sphere of radius 0.3 at time 0. Using an internal function in *Mathematica*, the snapshots can be animated to create a movie of the tornado.

Program 2 is written to accommodate many parcels of fluids; The limitation on the number of parcels is a function of the amount memory available on the CPU. Each parcel is colored differently so mixing and transport of fluid particles in various regions in the domain can be monitored visually.

Program 2 can also be modified readily to study the influence of the parameters k and P on the behavior of the solutions. According to the analysis presented in Serrin [1], the Navier–Stokes equations (1) allow for at least three types of tornado–like behavior, a funnel of the type shown in Figure 2, a funnel that descends towards the boundary $z = 0$ before spreading out, and a third solution with properties similar to the first but with a tight, pencil–like shape. It is not clear at this point which of these solutions, if any, is a stable solution of the full Navier–Stokes equations. In the future, we will embark on the program of numerically studying the stability of the individual funnels by perturbing each steady–state solution and monitoring the asymptotic state of the perturbed solution.

3 Rayleigh–Bénard Flow

In Camassa and Wiggins [3], a model of chaotic advection is presented that is based on time–periodic perturbation of a standard stream function formulation of the Rayleigh–Bénard flow. Let ψ be defined by

$$\psi(x, z, t) = \sin(\pi x)\sin(\pi z) + \epsilon\cos(\omega t)\cos(\pi x)\sin(\pi z), \qquad (7)$$

where ϵ and ω are given constants. The first part of ψ, $\sin\pi x \sin\pi z$, is the standard stream function for the Rayleigh–Bénard while ϵ and ω are the amplitude and frequency of the perturbation, respectively. Physically, this stream function models the motion of fluid particles that are being exposed to a temperature gradient in the z direction and a mechanical periodic motion in the x direction. The primary questions have to do with the transport and mixing of fluids and the asymptotic state of the flow.

The equations of motion of an individual fluid particle are related to ψ through the relations

$$\frac{dx}{dt} = \frac{\partial\psi}{\partial z}, \quad \frac{dz}{dt} = -\frac{\partial\psi}{\partial x}. \qquad (8)$$

It is easy to show that when ϵ is zero a typical particle of fluid undergoes a periodic motion confined to a 1×1 cell defined by the particles initial position and no mixing occurs between neighboring cells. But when $\epsilon > 0$ and $\omega > 0$, a typical fluid particle may visit neighboring cells (see Figure 3) and a fluid particle from one cell could possibly enter a cell far away from its original cell. More importantly, at least from a mathematical point of view, particles of fluid that are originally located near one of the vertical boundaries of a cell have a tendency to react more drastically to the perturbation term in (7) than the particles that are located near the middle of a cell. Thus the rate of mixing of fluids is not only influenced by ϵ and ω, but also by the location of a fluid particle prior to being acted upon by the perturbation.

To obtain a figure such as Figure 3 one must solve the system of differential equations in (7) for large values of t. Program 3 exhibits the code used in obtaining this figure, whose significant part is the block `myNDSolve`, designed especially for solving systems of differential equations over large time intervals while making efficient use of the built–in adaptivity of *Mathematica*'s `NDSolve`.

Program 3:

```
myNDSolve[f_,g_,initial_,
       tfinal_,deltat_]:=
 Block[{a,b,sol,sol1},
  a=initial[[1]]; b=initial[[2]];
  Do[sol1[i]=NDSolve[{
   x'[t]==f[x[t],z[t],t],
   z'[t]==g[x[t],z[t],t],
   x[(i-1)*deltat]==a,
   z[(i-1)*deltat]==b},
   {x,z}, {t,(i-1)*deltat,
       i*deltat}];
  graph[i]=ParametricPlot[
  Evaluate[{x[t], z[t]}/. sol1[i]],
    {t, (i-1)*deltat,i*deltat},
      DisplayFunction->Identity];
    a=First[x[i*deltat]/.sol1[i]];
    b=First[z[i*deltat]/. sol1[i]],
    {i, 1, tfinal/deltat}]
 ];

tfinal = 100;eps=0.1;
omega=6;deltat=1/10;
ep=ToString[eps];
om=ToString[omega];
time=ToString[tfinal];
llabel=StringJoin["eps = ",ep,",
    omega =",om,", t = ",time];
psi[x_, z_,t_] =
  Sin[Pi*x]*Sin[Pi*z]+
  eps*Cos[omega*t]*
  Cos[Pi*x]*Sin[Pi*z];
f[x_,z_,t_]=D[psi[x,z,t],z];
g[x_,z_,t_]=-D[psi[x,z,t],x];
initial={0.15,0.01};

solution=myNDSolve[f,g,
   initial,tfinal,deltat];

Show[Table[graph[i],
 {i, 1, tfinal/deltat}],
   DisplayFunction->$DisplayFunction,
     PlotLabel ->llabel]
```

An important piece of information in the analysis of (7) is how long it takes for the trjectory of a typical fluid particle to leave its original cell. In the case of fluid particle originally located in the "first" cell with $0 < x_0 < 1$, obtaining this information requires determining t^* and t^{**} such that

$$x(t^*) = 1 \quad \text{and} x(t^{**}) = 0.$$

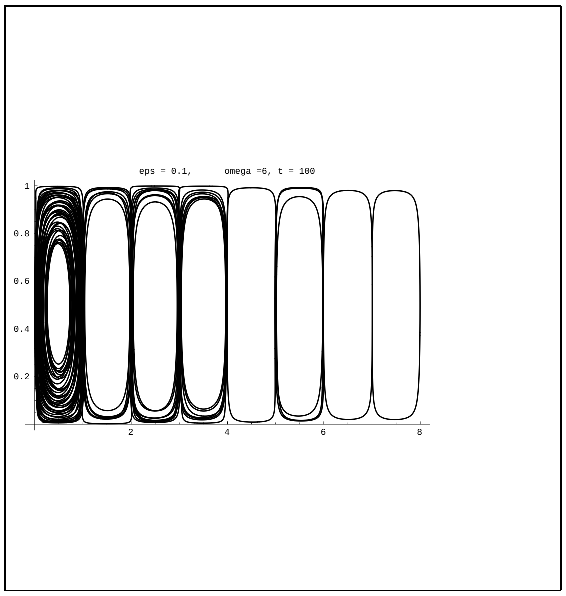

Figure 3: The trajectory of the particle located originally at $(0.15, 0.01)$. Here $\epsilon = 0.1$, $\omega = 6$ and $t \in (0, 100)$. This particle spends quite a bit of time in its original cell while also spending a substantial amount of time visiting other cells.

Since $x(t)$ is a numerical solution of a system of differential equations, determining these values of t requires applying `NDSolve` together with a root finder in *Mathematica*. Program 4 displays one such program.

Program 4:

```
eps=0.1;omega=6;
psi[x_, z_,t_] =
  Sin[Pi*x]*Sin[Pi*z]+
   eps*Cos[omega*t]*
    Cos[Pi*x]*Sin[Pi*z];
f[x_,z_,t_]=D[psi[x,z,t],z];
g[x_,z_,t_]=-D[psi[x,z,t],x];

sol1[a_, b_,tfinal_]:=
 NDSolve[{x'[t]==f[x[t],z[t],t],
    z'[t]==g[x[t],z[t],t],
    x[0]==a, z[0]==b}, {x,z},
    {t,0,tfinal}, MaxSteps->5000];
rightexit={};leftexit={};
Do[a=0.01+i/20; b=0.01;
   solution=sol1[a,b, 40];
   aa=ToString[a]; bb=ToString[b];
llabel=StringJoin[
    "(",aa,",",bb,")"];
Clear[c,average,x1, domain,
     range,tpositive,tnegative];
x1[t_]=First[x[t]-1.000001/.
   solution];
domain=Table[t,{t,0,40, 0.1}];
range=x1[domain];
testrange=Table[range[[i]]*
       range[[i+1]],
   {i, Length[range]-1}];
Catch[Do[index=i;
  If[testrange[[i]]<0, Throw[i]],
       {i, Length[testrange]}]];
 If[range[[index]]<0,
  tnegative=domain[[index]],
   tnegative=domain[[index+1]]];
 If[range[[index]]<0,
   tpositive=domain[[index+1]],
     tpositive=domain[[index]]];
 Do[average=
  (tpositive+tnegative)/2;
  c=x1[average];
   If[c < 0, tnegative=average,
    tpositive=average],{i, 50}];
 ExitToRight=average;
 rightexit=Append[rightexit,
   {a, b, ExitToRight}];
 x2[t_]=First[x[t]-0.00001/.
    solution];
 Plot[x2[t], {t, 0, 40},
   PlotPoints->500,
   PlotLabel->StringJoin["x(t)
         with x(0) = ", aa]];
 domain=Table[t,{t,0,40, 0.1}];
 range=x2[domain];
 testrange=Table[range[[i]]*
  range[[i+1]],
    {i, Length[range]-1}];
 Catch[Do[index=i;
  If[testrange[[i]]<0, Throw[i]],
   {i, Length[testrange]}]];
 If[range[[index]]<0,
   tnegative=domain[[index]],
   tnegative=domain[[index+1]]];
 If[range[[index]]<0,
   tpositive=domain[[index+1]],
   tpositive=domain[[index]]];
 Do[average=(tpositive+
    tnegative)/2;
   c=x2[average]; If[c < 0,
       tnegative=average,
     tpositive=average],{i, 50}];
   ExitToLeft=average;
   Print[llabel," exits to
     the cell
    on the right at t = ",
     ExitToRight];
   Print[llabel," exits to
     the cell
    on the left at t = ",
     ExitToLeft];
   leftexit=Append[leftexit,
     {a, b, ExitToLeft}],
{i, 1, 19}];
ExitTime=Table[Min[
     rightexit[[i,3]],
 leftexit[[i,3]]],
   {i,Length[leftexit]}];
graph1=ListPlot[ExitTime,
     PlotJoined->True,
     PlotRange->All,
     AxesOrigin->{0,0},
     PlotLabel-> StringJoin[
      "Exit time of
       particles located at z = ",
       bb]]
```

Figure 4 shows part of the output of this program. Note, in particular, that two nearby particles, originally located at $(0.06, 0.01)$ and $(0.11, 0.01)$, have remarkably different trajectories, alerting the reader to the chaos investigated in [3].

Program 4 combines `NDSolve` with the bisection method. It is written with the anticipation that after 40 units of time a particle will exit its cell either from the left or the right. Obvious modifications need be made if a particle does not exit its cell after a specified period of time. This program is also capable of displaying the graph of exit–time versus the horizontal coordinate of the original position of a set of particles chosen in a specific cell.

As a further testimony to the chaotic nature of the solutions of (8), we now display the evolution of a parcel of fluid much in the same spirit as was carried out in Program 2.

Program 5:

```
tfinal = 10;eps=0.1;n=20;
tinterval=1/50;
```

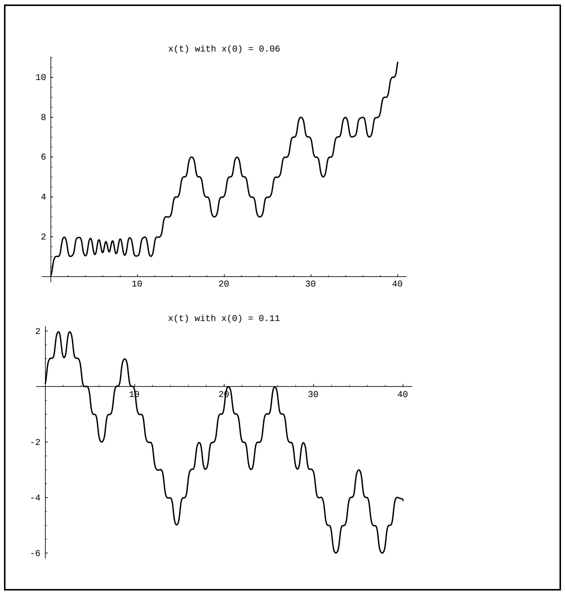

Figure 4: The trajectories of two particles located originally at $(0.06, 0.01)$ and $(0.11, 0.01)$. Again, $\epsilon = 0.1$ and $\omega = 6$. Program 4 determines whether each particle first exits its cell from the left or the right. This program is based on combining a differential equation solve and the bisection method as a root finder.

```
omega=6;
ep=ToString[eps];
om=ToString[omega];
psi[x_,z_,t_]=Sin[Pi*x]*Sin[Pi*z]+
 eps*Cos[omega*t]*Cos[Pi*x]*Sin[Pi*z];
f[x_,z_,t_]=D[psi[x,z,t],z];
g[x_,z_,t_]=-D[psi[x,z,t],x];
diffeqn[tfinal_, {b_, c_}]:=
 NDSolve[{x'[t]==f[x[t],z[t],t],
  z'[t]==g[x[t],z[t],t],
  x[0]==b, z[0]==c},
  {x,z}, {t,0,tfinal}];
InitialPoints=Table[
   {0.8 +0.1 Cos[t], 0.2 +0.1 Sin[t]},
{t, 0, 2 Pi, 2Pi/n}];
plot[1]=Show[Graphics[
  Table[Point[InitialPoints[[i]]],
       {i,Length[InitialPoints]}]],
   PlotRange->{{-1,1},{-1,1}},
      Axes->True,
         AspectRatio->Automatic];
oldsolution= Flatten[Table[
  diffeqn[tfinal,InitialPoints[[i]]],
    {i,1,Length[InitialPoints]}],1];
Do[currentTime=j*tinterval;
  sol=Table[{x[t],z[t]}/. o
  ldsolution[[i]]/. t->currentTime,
          {i,1, Length[oldsolution]}];
  points2 = Table[Point[sol[[i]]],
    {i, Length[sol]}];
  graph=Graphics[points2];
  time=ToString[currentTime];
  llabel=StringJoin[" t = ",time];
  plot[j+1]=Show[graph,
     PlotRange->{{-1,1},{-1,1}},
PlotLabel->llabel, Axes->True, A
 spectRatio->Automatic],
{j, 1, 100}]
graph3b=Show[GraphicsArray[
  {{plot[1]}, {plot[11]},
   {plot[51]}, {plot[81]}}]];
```

Figure 5 shows part of the output of Program 5. With $\omega = 6$ and $\epsilon = 0.1$, this program follows the evolution of 50 particles that formed a circle of radius 0.1, centered at $(0.2, 0.2)$ at time zero. The program then draws a snapshot of the location of these particles at time intervals of 0.1. Altogether one hundred snapshots are drawn, four of which are displayed in Figure 5. This figure clearly points to the chaotic charac-

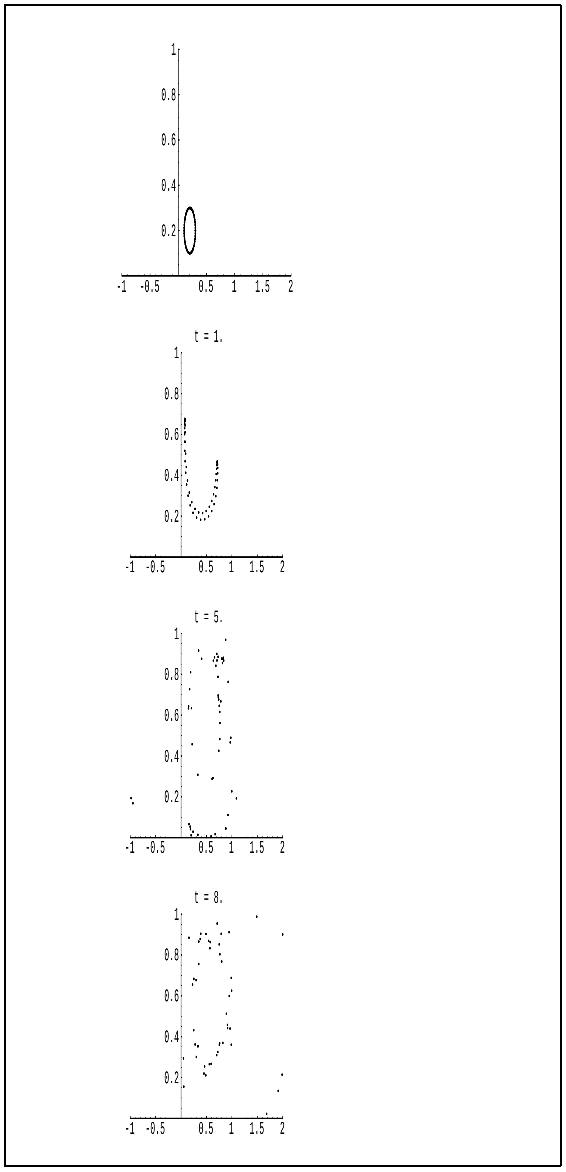

Figure 5: This figure shows four snapshots corresponding to the position of 50 particle points that originally formed a circle of radius 0.1 centered at $(0.2, 0.2)$. The times of the snapshots are $t = 0, 1, 5$ and 8. Note that, in a manner that is hard to predict, various particles have begun visiting neighboring cells by the time t has reached 8 while other particles have remained in their original cell.

ter of this flow since particles that at time zero were are at most 0.4 units apart are vastly separated by the time t reaches 10.

An additional technical tool that can be used in this figure is color. By using different colors for particles occupying various parts of the original circle, one may gain insight into which particles have more of a tendency to leave their cell and mix with particles in other cells.

4 Conclusions

In this paper we have discussed in some detail a numerical approach for studying sets of differential equations that govern two fundamental flows in fluid dynamics, tornadoes as steady–state solutions of the Navier–Stokes equations and the Rayleigh–Bénard problem. The common feature in both models is that because of the nature of the mathematical questions under investigation, we are obliged to use differential equation solvers in combination with root-finding techniques. These algorithms are available in *Mathematica*, as well as in a variety of other powerful software packages on the market. What should be emphasized here is how one is able to take the output of one internal function, such `NDSolve`, and make it available as input for another internal function, such as `FindRoot`. Programs 1 and 4 succeed in demonstrating this capability.

The main strategy of this paper, namely, first seeking the velocity field of a fluid flow as a solution of a partial differential equation and second, exhibiting trajectories of the resulting dynamical system, may be applied to a variety of fundamental physical problems, among which is the flow past an aircaft wing. We have succeeded in implementing this strategy for the so–called “vortex lattice method” (see [4]). In this method, in contrast with the Rayleigh–Bénard problem, the velocity field has a potential which is computed by applying the Biot–Savart formula to a lattice discretization of the wing. Once the potential is determined, the process of obtaining particle trajectories follows closely the algorithm described in Program 2 and 3.

A significant feature demonstrated in Program 1 is the successful implementation of the

Picard iteration scheme. Using this scheme we are able to reduce a nonlinear system of integro–differential equations to a sequence of ordinary differential equation, each of which lends itself to `NDSolve` of *Mathematica*. For the range of parameters P and k under consideration here, it is not difficult to show that the sequence of solutions of the differential equations converges, and that the limiting function is a solution of the original integro–differential system.

The Picard iteration scheme is a standard method in mathematics for obtaining solutions to nonlinear problems, especially for the case of nonlinear integral and partial differential equations. Several examples of this method when applied to problems of interest in ocean dynamics appear in [2]. We are planning on presenting applications of this method to the dynamics of the Gulf Stream as well as the unsteady Navier–Stokes equations in the near future.

5 Acknowledgements

Reza Malek–Madani gratefully acknowledges receiving partial support from the Office of Naval Research, grant number N0001497WR20002, while working on this project.

6 References

[1] Serrin, James, “The swirling vortex”, *Philosophical Transactions of the Royal Society of London* 271A (1972): 325 – 360.

[2] Malek–Madani, Reza, **Advanced Engineering Mathematics with** *Mathematica* **and** *MATLAB*, Volumes I and II, Addison–Wesley–Longman, Newton, MA, 1998.

[3] Camassa, R. and Wiggins, S., “Chaotic advection in a Rayleigh–Bénard flow”, *Physical Review A*, Vol 43, no. 2 (1991): 774 – 797.

[4] Bertin, John J. and Smith, Michael L., **Aerodynamics for Engineers**, 3rd Edition, Prentice Hall, Upper Saddle River, NJ, 1998.